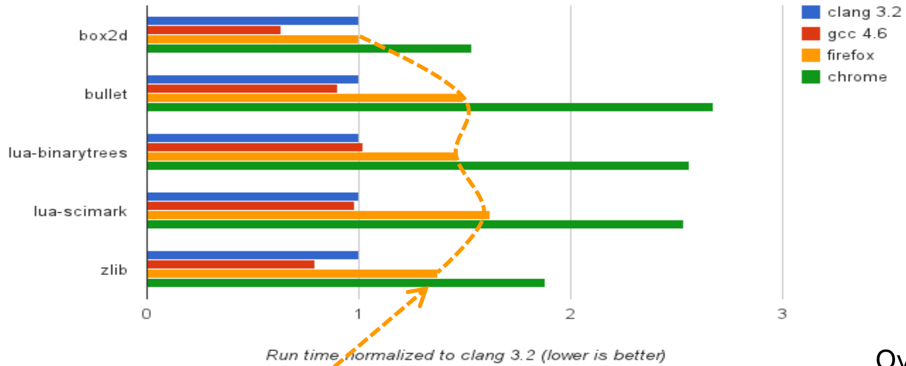
Bringing the Full Power of Modern Hardware to the Open Web Platform

Mohammad Reza Haghighat
Senior Principal Engineer, Intel Corporation

October 29, 2014

Astounding JavaScript* Performance Improvements



Very efficient code generated by Firefox* JIT

LLVM Bitcode

Emscripten

JavaScript*
asm.js

Epic* Games Unreal Engine* 3

http://www.unrealengine.com/html5/

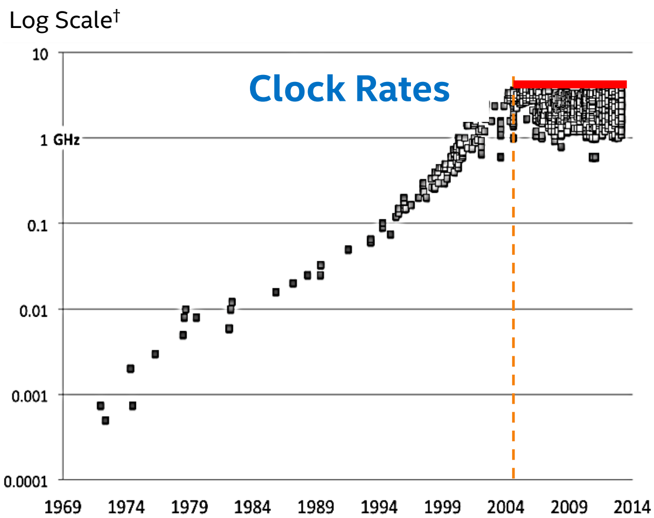
Over 1M lines of C/C++ code compiled to JavaScript* by Mozilla* and Epic

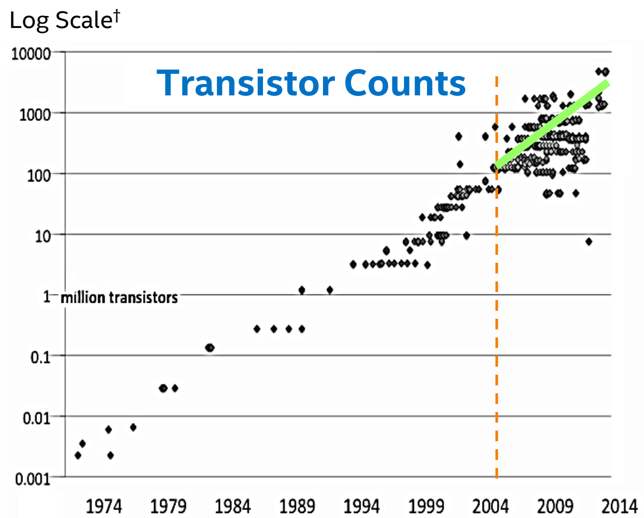
Achieving ~ 1.5x native running time via targeting asm.js†, a highly optimizable subset of JavaScript

JavaScript performance is approaching native speeds

† Courtesy of Mozilla Alon Zakai & Luke Wagner: http://people.mozilla.org/~lwagner/gdc-pres/gdc-2014.html#/

Microprocessor Trends – “Free Lunch” is over!





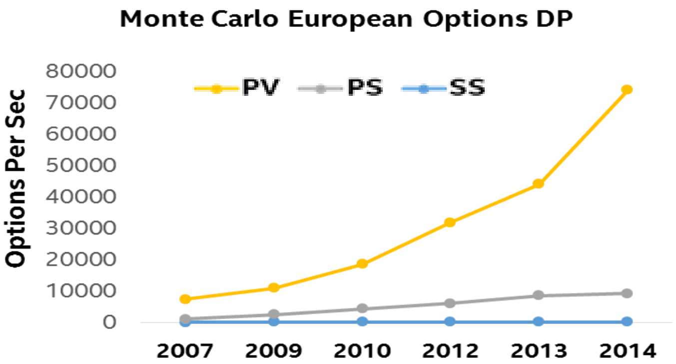
- Growth in processor clock rate halted around 2005
- Transistors per processor continues to grow exponentially

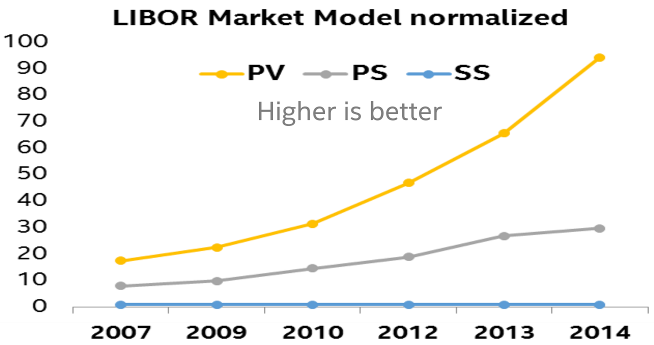
But, Moore’s Law continues with a shift to parallelism

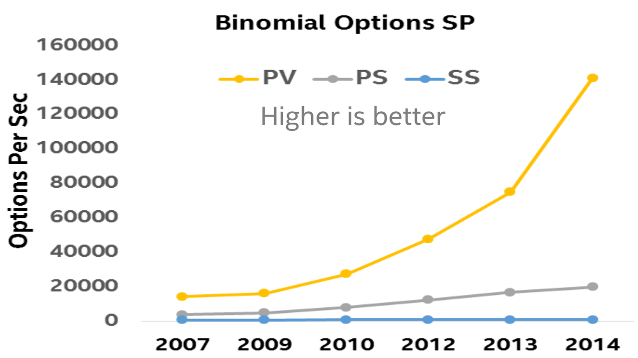
† (c) 2013, James Reinders and Jim Jeffers: *Intel® Xeon Phi™ High-Performance Programming*, used with permission.

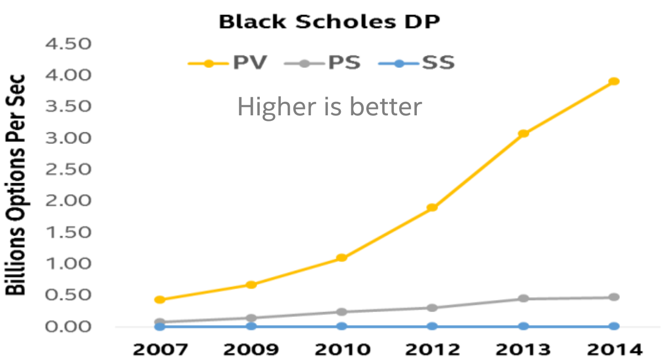
Parallelism is now Required to Benefit from Moore's Law

SS: Sequential Scalar **PS:** Parallel Scalar **PV:** Parallel Vector





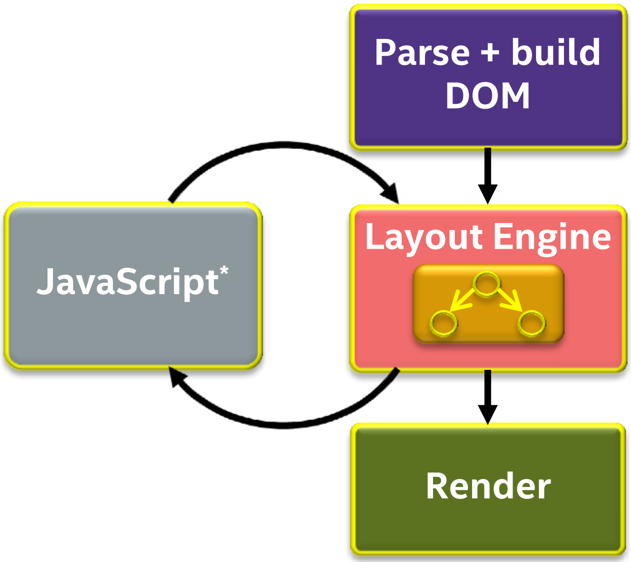


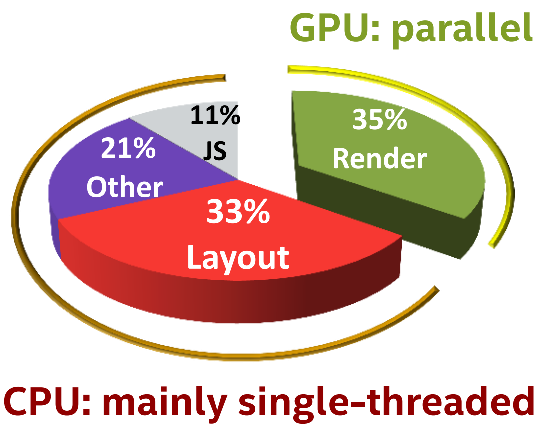


Open web client platform needs to be on Moore's Law curve

† Courtesy of Intel® Robert Geva: & Jim Jeffers: https://intel.activeevents.com/sf14/connect/sessionDetail.ww?SESSION_ID=1187

Optimizing Web Runtimes for Parallelism

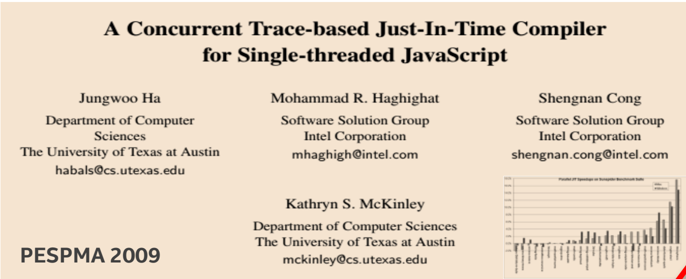




- HTML5 runtimes of today are not scalable with number of cores
- Need parallelism for both responsiveness and energy efficiency

Web runtimes need to be parallel end-to-end

Parallel Parsing and Compilation



A Concurrent Trace-based Just-In-Time Compiler for Single-threaded JavaScript

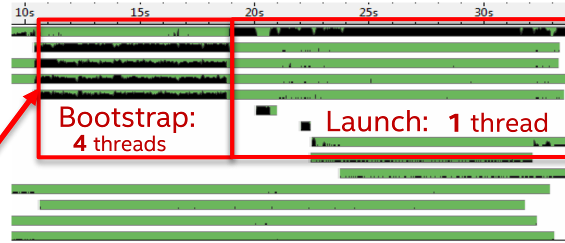
Jungwoo Ha
Department of Computer Sciences
The University of Texas at Austin
habals@cs.utexas.edu

Mohammad R. Haghighat
Software Solution Group
Intel Corporation
mhaghigh@intel.com

Shengnan Cong
Software Solution Group
Intel Corporation
shengnan.cong@intel.com

Kathryn S. McKinley
Department of Computer Sciences
The University of Texas at Austin
mckinley@cs.utexas.edu

PESPMA 2009

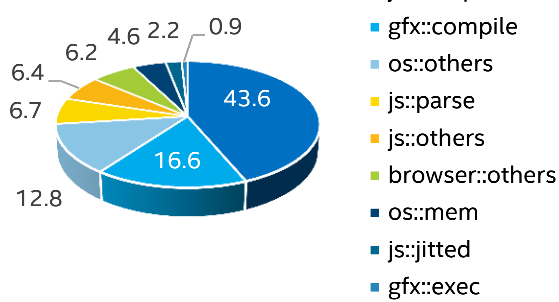


Epic* Citadel* profile on Firefox*

Process / Thread / Module / Function / Call Stack	CPU_CLK_UNHALTED.THREAD
firefox.exe	100.0%
Thread (0x11c0) JS and GFX execution	32.9%
Thread (0xfc)	19.1%
Thread (0x230)	14.5%
Thread (0xa00)	14.4%
Thread (0x148)	14.2%
Thread (0x480)	1.6%
Thread (0x10ec)	1.0%
Thread (0xf48)	1.0%
Thread (0x434)	0.7%
Thread (0x7c8)	0.3%
Thread (0x1344)	0.1%

Four threads for JavaScript* parsing and compilation

Cycle Breakdown



Background JIT compilers now in Chrome*, Firefox, Internet Explorer*, Safari*

SIMD – Single Instruction, Multiple Data

Scalar Operation

$A_x + B_x = C_x$

$A_y + B_y = C_y$

$A_z + B_z = C_z$

$A_w + B_w = C_w$

SIMD Operation of Vector Length 4 †

$$\begin{bmatrix} A_x \\ A_y \\ A_z \\ A_w \end{bmatrix} + \begin{bmatrix} B_x \\ B_y \\ B_z \\ B_w \end{bmatrix} = \begin{bmatrix} C_x \\ C_y \\ C_z \\ C_w \end{bmatrix}$$

SIMD operations deliver great performance & power efficiency

† Intel® Architecture currently has SIMD operations of vector length 4, 8, 16

Bringing SIMD to JavaScript*

Collaborators: Intel, Mozilla*, Google*, Microsoft*, ARM*, …

SIMD Number: http://wiki.ecmascript.org/doku.php?id=strawman:simd_number
Authors: Google's John McCutchan and Intel's Peter Jensen

Polyfill API: https://github.com/johnmccutchan/ecmascript_simd

float32x4, int32x4, Float32x4Array, Int32x4Array

```
var a = SIMD.float32x4 (1.0, 2.0, 3.0, 4.0);
var b = SIMD.float32x4 (5.0, 6.0, 7.0, 8.0);
var c = SIMD.float32x4.add (a, b);
```

Constructors: float32x4(x,y,z,w) float32x4.splat(s) float32x4.zero()

Operations: abs, neg, add, sub, mul, div, clamp, min, max, reciprocal, reciprocalSqrt, scale, sqrt, shuffle, shuffleMix, equal, notEqual, lessThan, greaterThan , withX, withY …

Status: In Firefox* Nightly, prototyped in Chromium*, on IEBlog roadmap£

1st stage approval for inclusion in ES7 by TC39†

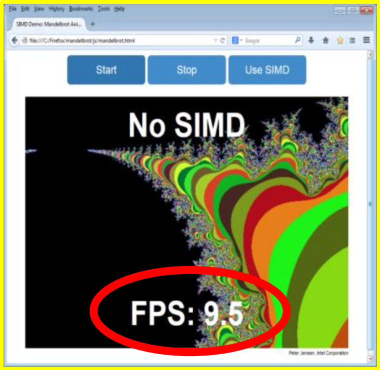
† A copy of the TC39 Presentation: http://esdiscuss.org/notes/2014-07/simd-128-tc39.pdf

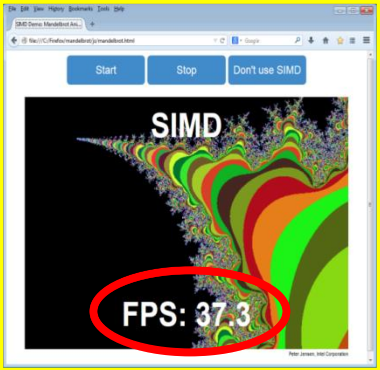
£ IEBlog: http://blogs.msdn.com/b/ie/archive/2014/09/18/updates-to-our-platform-roadmap.aspx

SIMD.JS – The API

```
function mandelx4(c_re4, c_im4) {
  var z_re4  = c_re4;
  var z_im4  = c_im4;
  var four4  = SIMD.float32x4.splat (4.0);
  var two4   = SIMD.float32x4.splat (2.0);
  var count4 = SIMD.int32x4.splat (0);
  var one4   = SIMD.int32x4.splat (1);

  for (var i = 0; i < max_iterations; ++i) {
    var z_re24 = SIMD.float32x4.mul (z_re4, z_re4);
    var z_im24 = SIMD.float32x4.mul (z_im4, z_im4);

    var mi4    = SIMD.float32x4.lessThanOrEqual (SIMD.float32x4.add (z_re24, z_im24), four4);
    // if all 4 values are greater than 4.0, there's no reason to continue
    if (mi4.signMask === 0x00) {
      break;
    }

    var new_re4 = SIMD.float32x4.sub (z_re24, z_im24);
    var new_im4 = SIMD.float32x4.mul (SIMD.float32x4.mul (two4, z_re4), z_im4);
    z_re4       = SIMD.float32x4.add (c_re4, new_re4);
    z_im4       = SIMD.float32x4.add (c_im4, new_im4);
    count4      = SIMD.int32x4.add (count4, SIMD.int32x4.and (mi4, one4));
  }
  return count4;
}
```

Our Firefox* Prototype



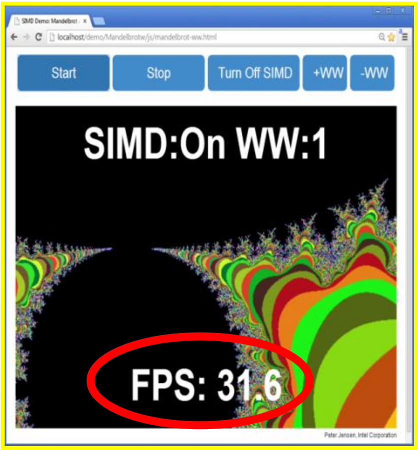


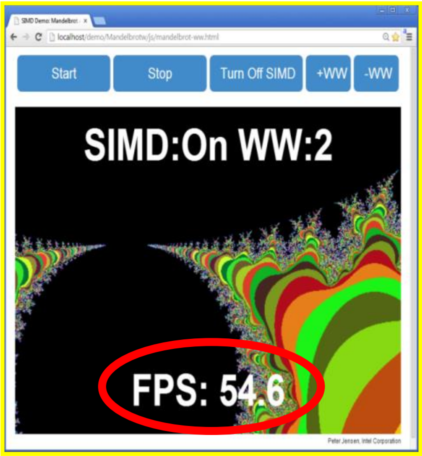
Our SIMD prototype delivers 3x~4x Mandelbrot speedup†

† Initial support for float32x4 and int32x4

Combining SIMD and Higher-Level Parallelism

WW: Number of WebWorkers



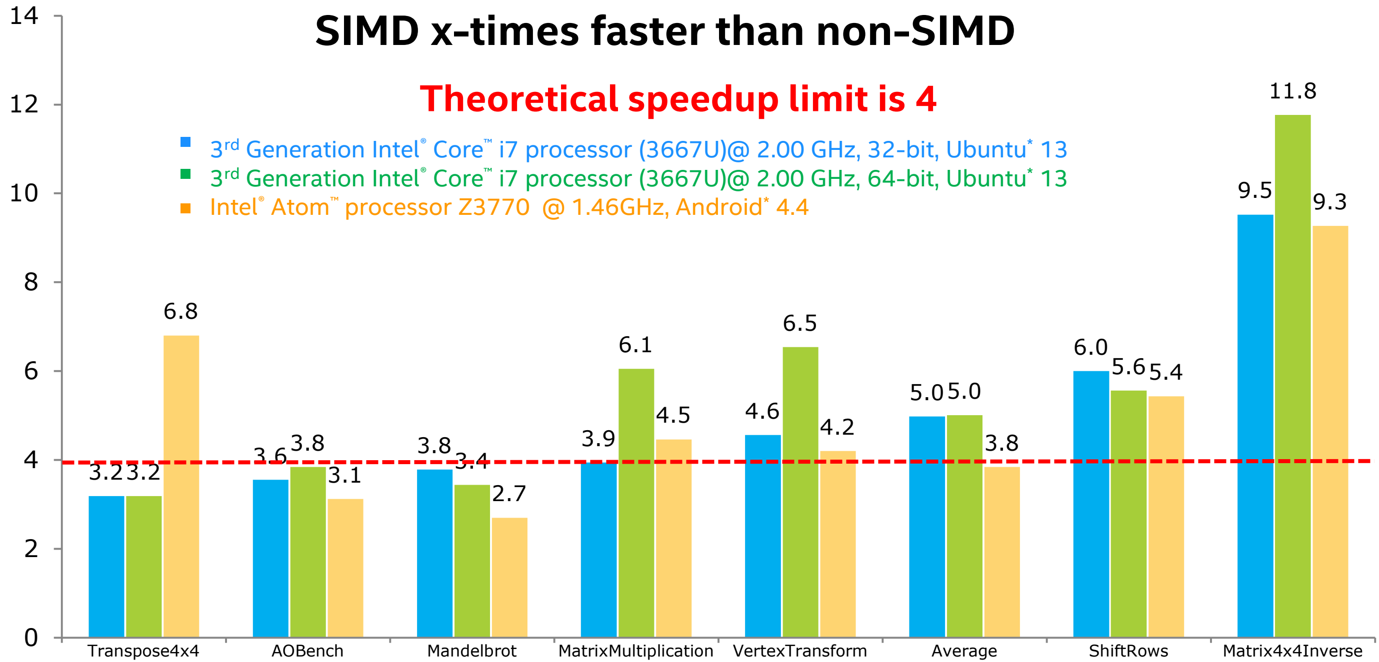


Our Chromium* Prototype

SIMD speedup is nicely multiplied by WebWorkers†

† Source: Intel® Peter Jensen : https://github.com/PeterJensen/
SIMD.JS demos: http://peterjensen.github.io/idf2014-simd

SIMD Speedups on Chromium*



Excellent early results while still focused on functionality

SIMD.JS benchmarks: https://github.com/johnmccutchan/ecmascript_simd/tree/master/src/benchmarks

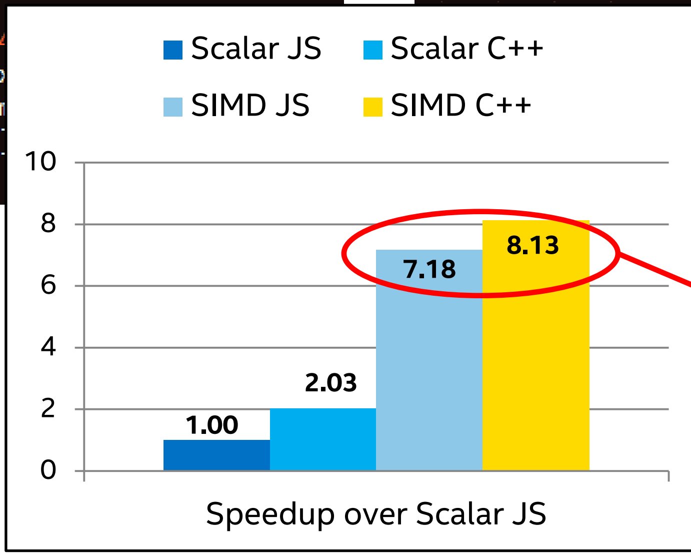
Emscripten now targets SIMD.JS

```
float simdAverage(float *src, int len) {
  __m128 sumx4 = _mm_setzero_ps();
  for (int i = 0; i < len; i+=4) {
    __m128 v = _mm_load_ps(src + i);
    sumx4 = _mm_add_ps(sumx4, v);
  }
  float sumx4_mem[4
  _mm_store_ps(sumx
  return (sumx4_mem
     + sumx4_mem[2]
}
```

C/C++

```
$16 = HEAP32[$src >> 2] | 0;
$i$02$i = 0;
$sumx4$01$i = SIMD.float32x4.splat(0);
$17 = $16 >> 4;
do {
 $sumx4$01$i = SIMD.float32x4.add($sumx4$01$i,
                   32x4[$17 + ($i$02$i >> 4)]);
                    + 4 | 0;
                   | 0) < 1e4);
                   Ptr >> 3] =
                    ($sumx4$01$i.z +
                    $sumx4$01$i.y))) / 1.0e4;
```

JavaScript*



Near-native SIMD.JS speedup

Emscripten brings native SIMD apps to the open web platform

Toward Perceptual Computing†

Learning & Education

Immersive Collaboration

3D Scanning and Sharing

Scan it

Share it

Customize and Print it

Gaming

Speech

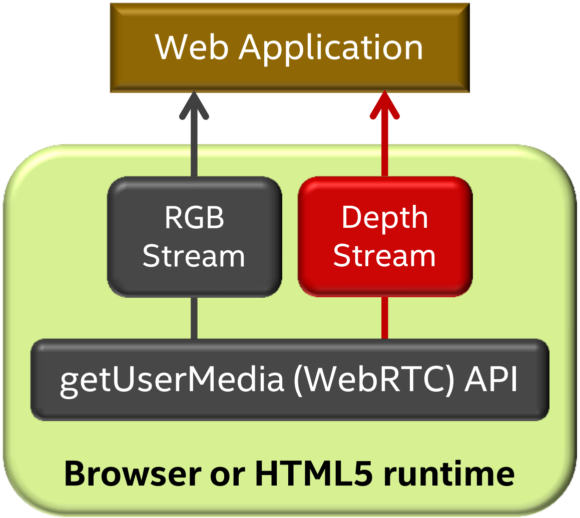
Out-of-reach Device Input

Devices sense and perceive user actions in a natural way

† Source: Intel® Perceptual Computing SDK: www.intel.com/software/perceptual

3D Cameras Make Perceptual Computing Accessible

Embedded 3D Camera

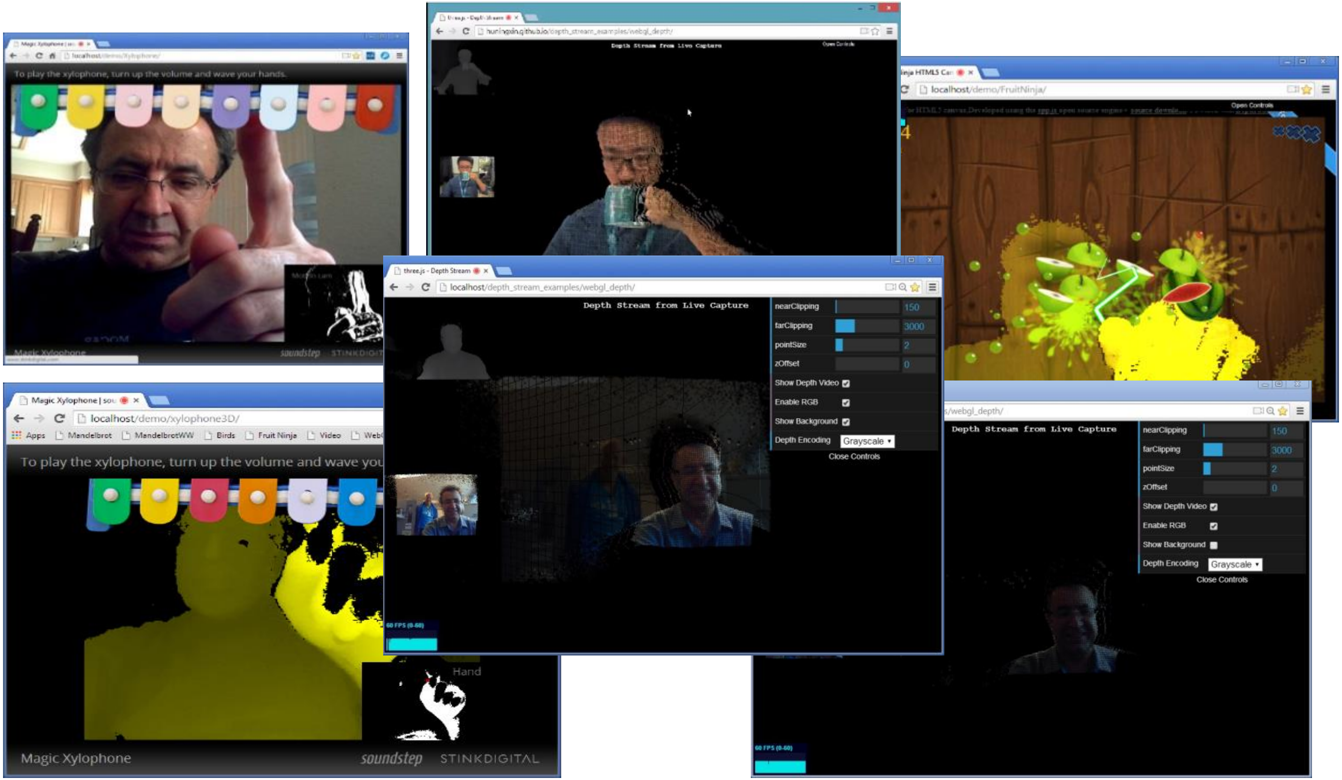



```
navigator.getUserMedia
  ({ video: true, depth: true }, success, failure);

function success(s) {

  var video = document.querySelector('#video');
  video.src = URL.createObjectURL(s);
  video.play();

  // construct MediaStream from the existing depth track(s)
  var depthStream = new MediaStream(s.getDepthTracks());

  // send the created depth stream over a RTCPeerConnection
  var peerConnection = new RTCPeerConnection(config);
  peerConnection.addStream(depthStream);

  var depthVideo = document.querySelector('#depthVideo');
  depthVideo.src = URL.createObjectURL(depthStream);
  depthVideo.play();
}
```

Media Capture Depth Stream Extensions are in W3C WG†

† W3C Media Capture Depth Stream Extensions: http://w3c.github.io/mediacapture-depth/

Toward Perceptual Web†



† 3D Camera WebRTC Demos, Courtesy of Intel® Ningxin Hu: https://www.youtube.com/channel/UC3eppo33tlz_EP7NWtZc0jQ
Demo Sources: Intel® Ningxin Hu: https://github.com/huningxin/depth_stream_examples
WebRTC Google* Code: http://webrtc.googlecode.com/svn/trunk/samples/js/demos/html/
Magic Xylophone: Soundstep*.com: http://www.soundstep.com/blog/experiments/jsdetection/

Web: The Most Viable Cross-Platform Technology Today

Visual, Perceptual, Full HW Access

and the Ubiquitous Application Platform of the Future

Legal Disclaimer

Risk Factors

The above statements and any others in this document that refer to plans and expectations for the second quarter, the year and the future are forward-looking statements that involve a number of risks and uncertainties. Words such as “anticipates,” “expects,” “intends,” “plans,” “believes,” “seeks,” “estimates,” “may,” “will,” “should” and their variations identify forward-looking statements. Statements that refer to or are based on projections, uncertain events or assumptions also identify forward-looking statements. Many factors could affect Intel’s actual results, and variances from Intel’s current expectations regarding such factors could cause actual results to differ materially from those expressed in these forward-looking statements. Intel presently considers the following to be important factors that could cause actual results to differ materially from the company’s expectations. Demand for Intel's products is highly variable and, in recent years, Intel has experienced declining orders in the traditional PC market segment. Demand could be different from Intel's expectations due to factors including changes in business and economic conditions; consumer confidence or income levels; customer acceptance of Intel’s and competitors’ products; competitive and pricing pressures, including actions taken by competitors; supply constraints and other disruptions affecting customers; changes in customer order patterns including order cancellations; and changes in the level of inventory at customers. Intel operates in highly competitive industries and its operations have high costs that are either fixed or difficult to reduce in the short term. Intel's gross margin percentage could vary significantly from expectations based on capacity utilization; variations in inventory valuation, including variations related to the timing of qualifying products for sale; changes in revenue levels; segment product mix; the timing and execution of the manufacturing ramp and associated costs; excess or obsolete inventory; changes in unit costs; defects or disruptions in the supply of materials or resources; and product manufacturing quality/yields. Variations in gross margin may also be caused by the timing of Intel product introductions and related expenses, including marketing expenses, and Intel's ability to respond quickly to technological developments and to introduce new products or incorporate new features into existing products, which may result in restructuring and asset impairment charges. Intel's results could be affected by adverse economic, social, political and physical/infrastructure conditions in countries where Intel, its customers or its suppliers operate, including military conflict and other security risks, natural disasters, infrastructure disruptions, health concerns and fluctuations in currency exchange rates. Intel’s results could be affected by the timing of closing of acquisitions, divestitures and other significant transactions. Intel's results could be affected by adverse effects associated with product defects and errata (deviations from published specifications), and by litigation or regulatory matters involving intellectual property, stockholder, consumer, antitrust, disclosure and other issues, such as the litigation and regulatory matters described in Intel's SEC filings. An unfavorable ruling could include monetary damages or an injunction prohibiting Intel from manufacturing or selling one or more products, precluding particular business practices, impacting Intel’s ability to design its products, or requiring other remedies such as compulsory licensing of intellectual property. A detailed discussion of these and other factors that could affect Intel’s results is included in Intel’s SEC filings, including the company’s most recent reports on Form 10-Q, Form 10-K and earnings release.

Rev.
4/15/14

intel®